Attorney's Docket No.: 10559-518001 / P12422

APPLICATION

FOR

UNITED STATES LETTERS PATENT

10081925.022002

TITLE: METHOD AND APPARATUS FOR PERFORMING A PIXEL AVERAGING INSTRUCTION

APPLICANT: YUYUN LIAO, NIGEL C. PAVER and JAMES E. QUINLAN

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL 688 322 755 US

I hereby certify under 37 CFR §1.10 that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

February 20, 2002
Date of Deposit

[illegible]
Signature

Gabriel Lewis
Typed or Printed Name of Person Signing Certificate

METHOD AND APPARATUS FOR PERFORMING A PIXEL AVERAGING INSTRUCTION

BACKGROUND

[0001] Many image and video processing techniques include operations in which a number of pixel values are averaged. These pixel averaging operations may be used for, for example, filtering and image estimation. The averaging may be performed on the pixel values of a number of neighboring pixels. The averaging may also be performed on the pixel values corresponding to the same pixel at different times, e.g., between frames. These averaging operations may be computationally intensive.

[0002] In order to support the computational load and data throughput requirements associated with performing a large number of averaging operations, processors used for image and video processing may introduce SIMD (Single-Instruction/Multiple-Data) operations. In SIMD operations, a single instruction is sent to a number of processing elements, which perform the same operation on different data.

[0003] One type of SIMD operation utilized in some image processing algorithms is a four-pixel averaging operation.

10031926.022002

A SIMD arithmetic logic unit (ALU) used to produce the average values may perform four addition and averaging operations on four sets of pixel values simultaneously to produce four 8-bit pixel average values. A 40-bit SIMD adder may be used to perform this instruction on 8-bit values. The 40-bit SIMD adder includes two dummy bits for each byte. One dummy bit controls the blocking or propagation of carries from an addition operation, and the other dummy bit controls a shifting operation. The shifting operation may be performed only in the four-pixel averaging operation, while other instructions that utilize the SIMD adder may only require a 36-bit adder. A 40-bit SIMD adder may have a larger layout than a 36-bit SIMD adder and require additional structure to accommodate the extra dummy bits, taking up limited chip area just to accommodate one instruction.

10081925.022002

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] Figure 1 is a block diagram of a functional unit for performing a four-pixel averaging SIMD (Single-Instruction/ Multiple-Data) operation according to an embodiment.

[0005] Figure 2 is a block diagram illustrating the bit positions of a SIMD adder according to an embodiment.

[0006] Figure 3 is a block diagram illustrating the operational flow of a four-pixel averaging operation according to an embodiment.

DETAILED DESCRIPTION

[0007] Figure 1 illustrates a functional unit 100 for performing a four-pixel averaging SIMD (Single-Instruction/Multiple-Data) instruction according to an embodiment. The functional unit 100 may be implemented in a processor, for example, a general purpose processor, a digital signal processor (DSP), or and application specific integrated circuit (ASIC) processor, for image and/or video processing.

[0008] The four-pixel average (FPA) SIMD instruction be performed on the pixel values of four neighboring pixels and/or the pixel values of the same pixel at different times, e.g., between frames. The FPA SIMD instruction may treat the pixel values (operands) as packed 8-bit (byte) values.

[0009] The FPA instruction may be implemented using the following format:

FPA <H,L> {R} wRd wRn wRm,

10081525.022502

where H, L, and R are qualifiers, wRm and wRn are packed operands, and wRd is a destination register 102. The qualifier H indicates that the results are to be placed in the high order bytes 103 of the destination register (wRd) 102, The qualifier L indicates that the results are to be placed in the low order bytes 104 of the destination register 102, and R is a rounding value, which may be set to a value of 2_{10} (10_2). The values of wRm and wRn and wRd may be treated as unsigned, packed 8-bit data and the results of the FPA SIMD instruction may be written in unsigned, packed 8-bit format.

[0010] The four-pixel averaging operation may be carried out as follows:

```
If (H Specified) then
  wRd[byte7]=(wRn[byte 4]+wRm[byte 4]+wRn[byte 3]+wRm[byte 3]+Round)>>2
  wRd[byte6]=(wRn[byte 3]+wRm[byte 3]+wRn[byte 2]+wRm[byte 2]+Round)>>2
  wRd[byte 5]=(wRn[byte 2]+wRm[byte 2]+wRn[byte 1]+wRm[byte 1]+Round)>>2
  wRd[byte 4]=(wRn[byte 1]+wRm[byte 1]+wRn[byte 0]+wRm[byte 0]+Round)>>2
  wRd[byte 3] = 0
  wRd[byte 2] = 0
  wRd[byte 1] = 0
  wRd[byte 0] = 0;
Else if (L Specified) then
  wRd[byte 7] = 0
  wRd[byte 6] = 0
  wRd[byte 5] = 0
```

wRd[byte 4] = 0

wRd[byte 3]=(wRn[byte 4]+wRm[byte 4]+wRn[byte 3]+wRm[byte 3]+Round)>>2

wRd[byte 2]=(wRn[byte 3]+wRm[byte 3]+wRn[byte 2]+wRm[byte 2]+Round)>>2

wRd[byte 1]=(wRn[byte 2]+wRm[byte 2]+wRn[byte 1]+wRm[byte 1]+Round)>>2

wRd[byte 0]=(wRn[byte 1]+wRm[byte 1]+wRn[byte 0]+wRm[byte 0]+Round)>>2

where ">>2" indicates that the result of the addition operation is shifted right by two bits.

[0011] A set of five operands, wRn[byte 0] to wRn[byte 4], are stored in a wRn register 105. Another set of five operands, wRm[byte 0] to wRm[byte 4], are stored in a wRm register 106. A compressor stage 108 includes four "5-to-2" compressors 110-113. Each compressor 110-113 compresses five vectors, i.e., four operands and the rounding value, into two vectors. For example, compressor 110 receives the vectors wRn[byte 4], wRm[byte 4], wRn[byte 3], wRn[byte 3], and the rounding value ($R = 2_{10}$), and generates a sum vector (S) and a carry vector (C). The sum and carry vectors generated by compressor 110 may be passed to a 36-bit SIMD (Single-Instruction/Multiple-Data) adder 114 along with the sum and carry vectors generated by the other compressors 111-113.

[0012] The SIMD adder 114 operates on the various sum and carry vectors from the different compressor 110-113 separately to produce four 8-bit pixel average value results. The SIMD adder 114 may be 36-bits wide, including a dummy bit 202-205 for each of the four byte locations 206-209, as shown in Figure 2. The dummy bits block or propagate the carries from the addition operation performed by the SIMD adder. The 36-bit SIMD adder may also be used by other instructions, and may operate on packed 8-bit, packed half word (16-bit), and packed word (32-bit) operands.

[0013] The results in the byte locations 206-209 output from the SIMD adder 114 are directed to the proper byte locations in the destination register 102 by a multiplexer 116 in response to a select signal 118. If H is selected, the four pixel average values are placed in high order byte positions 103 (wRd[byte 4]…wRd[byte 7]). Otherwise, the four pixel average values are placed in low order byte positions 104 (wRd[byte 0]…wRd[byte 3]).

[0014] As described above, the FPA instruction adds four pixel values wRm[byte i] + wRn[byte i] + wRm[byte (i-1) + wRn[byte (i-1)] for i = 1 $\rightarrow$ 4, and then produces an average value by dividing the sum by four. In binary division, dividing a number by four may be accomplished by adding a round value of 2_{10} (01_2) and shifting the result right by two bit positions. The two least significant bits (LSBs) of the result are discarded.

[0015] Typically, the compressors pass sum and carry vectors to a SIMD ALU which performs both an addition operation and a shifting operation. Figure 3 illustrates an operational flow performed by the 5-to-2 compressors 110-113 to produce a sum vector (S) and a carry vector (C) which can be added by the SIMD adder 114 to produce an pixel average value result directly, i.e., without the shifting operation.

[0016] Each 5-to-2 compressor 110-113 may include three stages of "3-to-2" compressors, a first carry shift adder (CSA) stage 302, a second CSA stage 304, and a third CSA stage 306. The 3-to-2 compressors each compress three vectors into a sum vector and a carry vector. The different 5-to-2 compressors 110-113 operate similarly, but on different pixel values. Consider the 5-to-2 compressor 110, which compresses the operands wRm[byte 4], wRn[byte 4], wRm[byte 3], wRn[byte 3], and the rounding vector R. In this case, WRm<0>…<7> correspond to the bits of wRm[byte 3], WRn<0>…<7> correspond to the bits of wRn[byte 3], WRm<8>…<15> correspond to the bits of wRm[byte 4], and WRn<8>…<15> correspond to the bits of wRn[byte 4]. In the first CSA stage 302, wRm[byte 3], wRn[byte 3], and wRm[byte 4] are compressed into a sum vector S0 having bits S0<0>…S0<7> and a carry vector C0 having bits C0<0>…C0<7>.

In the second stage, wRn[byte 4], S0, and a Round vector of 2_{10} (x<1>x<0> = 10_2) are compressed into a sum vector S1 having bits S1<0>…S1<7> and a carry vector C1 having bits C1<0>…C1<7>. In the third stage, vectors C0, S1, and C1 are compressed into a sum vector S2 having bits S2<0>…S2<7> and a carry vector having bits C2<0>…C2<7>.

[0017] As described above, the least two LSBs of the result of (wRn[byte 4] + wRm[byte 4] + wRn[byte 3] + wRm[byte 3] + Round) are discarded in the shifting operation (>>2). The only useful data from the two LSBs is the carry out C2<0>. Since the LSBs of the three input vectors C0, S1, and C1 in the third stage 304 of the operation 300 are 0, S1<0>, and 0, respectively, the carry out, C2<0>, equals 0.

[0018] In conventional implementations, 10-bit values {0 S2<8> … S<0>} and {C2<8> C2<7> … C2<0> 0} are fed into a 10-bit adder in a 40-bit SIMD ALU, and then the result is shifted right by 2-bits. Since the carry out C2<0> from last two bits is 0 and last 2 bits of the results from the adder are discarded anyway, only 8-bit values {0 S2<8> …S<3> S<2>} and {C2<8> C2<7> … C2<2> C2<1>} are needed. These 8-bit values may be added by an 8-bit adder in the 36-bit SIMD adder 114. The result from the addition operation performed by the adder 114 does not need to be shifted right by 2 bits. Thus, the 36-bit SIMD adder generates the four-pixel average values of the FPA instruction direct in one (addition) operation, and the adder does not need to perform the shifting operation.

[0019] The 40-bit SIMD ALUs that may be used to perform the FPA instruction include two dummy bits for each byte, one block or propagate carries, and the other to control the shifting operation. However, in general, other instructions do not require the shifting operation, and may be performed on the 36-bit SIMD adder 114, which includes only one dummy bit 202-205 per byte 206-209.

[0020] The 36-bit SIMD adder 114 may be desirable from a design and performance perspective over a comparable 40-bit SIMD ALU for several reasons. The 36-bit SIMD adder may have a shorter critical path delay and a layout which is >10% smaller than that of a comparable 40-bit SIMD ALU. Furthermore, in order to align with the 40-bit SIMD adder layout, all other functional units in the data path may have to accommodate one more dummy data bit for every byte. Consequently, the whole data path may have to be increased by >10% just to accommodate the FPA SIMD instruction.

[0021] Although packed 8-bit operands and results have been described, the functional unit 100 and FPA SIMD

10081492.022002

instruction may operate on other data sizes (e.g., 16-bit and 32-bit operands) by selecting component functional units and data paths layouts that accommodate the different data size. Also, SIMD instructions that operate on data multiples other than four, as described above in connection with the 36-bit adder, may also be implemented by selecting component functional units and data path layouts that accommodate the different data multiples.

[0022] A number of embodiments have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. Accordingly, other embodiments are within the scope of the following claims.